SOFTWARE INTERNSHIP REPORT

June 25, 2018

Samantha Tureski
Department of Linguistics
University of Tübingen
samantha-zoe.tureski@student.uni-tuebingen.de

Abstract

This paper reports a three-month internship at the Department of Linguistics (SfS) at Eberhard Karl University Tübingen. The research group that supported my internship, the Feedbook Project, is developing an interactive online workbook for English language learners at the Germany secondary school level. The Feedbook Project focuses its research on using computational methods to improve the learning experience. My task was to implement a tool to track students' usage of spelling and out of vocabulary (OOV) terms in their written homework assignments. This paper reports an overview of the tool's creation, from the back-end programming, to the integration of the tool into the user interface. Further internship tasks, such as training and assessing other tools, are also remarked upon.

1 The Project

The Feedbook Project is a joint undertaking between the University of Tübingen and the Diesterweg, an educational books publisher, for their Camden Town English Textbook series. It is funded through the Knowledge Transfer Project (T1) within the collaborative research center known as the Sonderforschungsbereich (SFB) 833. The project aims to create a fully-comprehensive online version of the workbook that accompanies the seventh-year *Camden Town* textbook, complete with individualized feedback and streamlined grading capabilities for teachers.

1.1 Background

In German secondary schools, English learning comprises an important part of a student's education. There has previously been some mixed findings on the necessity of overt error correction in L2 (second language) classrooms. Evidence shows that error correction makes little difference in L2 students' written progress [1]. It may also reinforce the belief that "it is the teacher's job to correct errors," thereby excusing students from responsibility for their own learning [2]. Other research, however, points to L2 learners' increased satisfaction and receptivity in regard to comprehensive teacher feedback [3].

2 The Problem

Four seventh-year L2 English classrooms from gymnasia around Tübingen agreed to participate in the Feedbook Project. In these classrooms, correction of student workbook exercises had previously remained a challenge. Student self-grading was only feasible for exercises with an answer key. For open-ended writing tasks, the process of collecting physical workbooks to correct individual assignments was cumbersome for instructors. Teachers also had no concrete overview of students' progress, and students lacked access to real-time hints and standardized feedback.

3 The Online Workbook

The Feedbook online workbook [4] contains a diverse sampling of 55 exercises in the seventh-year *Camden Town* English workbook. It is accessed by the means of a login page. Each user has his own account, with "Teacher" type accounts being able to create and access responses input from "Student" type accounts.

3.1 Exercise Types and Evaluation Methods

Just as in a real classroom, a teacher using the Feedbook online workbook has a full capacity to view and grade students' work. However, "Closed" types of exercises, such as matching, multiple choice, true or false, and fill-in-the-blanks, have only one correct answer and thus can be automatically corrected by Feedbooks internal correction tools.

In the backend, Feedbook is organized in a way that developers can choose exactly which question formats to add new features to. In the case of such aforementioned Closed-style tasks, a number of tools have been developed. Beyond simply grading, the interactive workbook will offer grammar corrections and hints for incorrect answers (e.g. "The number of this verb is wrong. Make sure it agrees with the subject.") There have been extensive efforts to collect and input data (both answers and corrections) from real student workbooks, so that the system can "learn" how to give feedback in a way that mimics a trained professional. A developer can query the similarly-organized database to test task-specific tools on real student data.

Freed from monotonous work of grading responses with an answer key, teachers have more time to focus on the style and content of student responses to open-ended tasks. Some exercises prompt students to respond to a short text by writing one sentence each for a series of questions (e.g. "How did [the character] feel at the end of the journey?"); these are denoted in the underlying database as Semi-Open tasks. Totally-Open exercises require an original piece of writing (e.g. "Describe this picture"). Even with Semi-Open and Open-style exercises, Feedbook makes every effort to assist teachers in discovering mistakes, thereby quickly passing meaningful feedback along to students.

3.2 Other Features

In addition to individualized feedback, the online workbook has a number of other capabilities:

- An internal mailing system by which instructors can send homework reminders and grading updates to individual students
- Statistical overviews for both class and student performance on an assignment or sets of assignments. Such overviews allow teachers to track progress and address common mistakes.
- Full multimedia integration of the audio and video required for some of the exercises

3.3 Community Involvement

Feedbook Project developers maintain constant contact with pilot classrooms. As well as the presence of a teacher from one of the classrooms at the projects weekly meeting, there are occasional developer classroom visits to test new features and consult with students.

4 The Internship

I was familiar with the Feedbook Project because I had previously held a research assistant position inputting raw workbook data to train the correction tagger. My existing understanding of the capabilities and weaknesses of the online workbook, combined with a desire for flexible working hours, made the Feedbook Project an ideal choice for my internship.

4.1 Tasks and Goals

The primary objective of my internship was to develop a spelling annotator in Java for Open-style exercises in the online workbook. But before I could begin work on this task, I spent a preliminary period familiarizing myself with the technologies used in the project. A third responsibility involved ascertaining the feasibility of integrating an open-source grammar-checking tool into the workbook.

4.1.1 The Training Period

One of the projects cornerstone technologies was Apache UIMA [5], a framework for creating pipelines for unstructured data. On top of this, the library uimaFit is used to simplify the testing process of UIMA components by automating the generation of certain descriptor files (which would otherwise have to be manually generated), as well as the passing in of data to a given pipeline [6]. DKPro Core is a set of natural language processing tools that have been built on the UIMA framework [7]. The online workbook already employed a number of DKPro components, such as a part-of-speech tagger, a lemmatizer, and a segmenter.

Before beginning work on the main codebase, I made my own toy UIMA pipeline project, in which I generated my own text-based object, called a JCas, that could be passed through the DKPro natural language processing tools. To visualize the tools output for a specific text, I learned how to access the UIMA Annotation Viewer, which creates a user-friendly, color-coded view of each tools input and output for every word in a given text document. A further step in my training was learning how to query the underlying database for certain types of pre-existing responses that had been typed directly from real classroom workbooks by research assistants like me. My intention was to eventually add my spelling annotation tool to the main project, and to then test it on many instances of real student data. Indeed, this initial pipeline project would serve as the basis for my internship- a safe environment in which I could efficiently test ideas before integrating them into the existing codebase.

In order to import and play with uimaFit and DKPro, it was necessary for me to gain understanding of Apache Maven, a tool which allows Java developers to keep a projects dependencies all in one place. It was necessary as well to spent some time reading the documentation for our version control system Subversion (SVN), and to become comfortable

with the Redmine, the projects online issue-tracking system. An interesting pairing feature of Subversion and Redmine is that issues on Redmine can be referenced and dynamically updated by adding issue IDs to commit messages in Subversion, allowing all colleagues to see current code progress on every reported issue.

4.1.2 Building the Out of Vocabulary Annotator

The primary goal of my internship was to build an Out of Vocabulary (OOV) Annotator. Beyond allowing instructors to see whether students were using vocabulary from sources outside the textbook, the annotator would have full spell checking capabilities for all in-vocabulary terms, as well as comprehensive advice for different types of misspellings. When I began, it was not yet clear how the tool would be used for students or for teachers, so I attempted to make a general Spelling Annotator that could benefit both types of users.

4.1.2.1 Creating a Digital Representation of the Textbooks Vocabulary

The Camden Town textbook and workbook were provided to me in both PDF and physical formats. If the Spelling Annotator were to check students usage of in-textbook vocabulary, I would need to collect all words from the glossary in the back of the book. This was a rather straightforward pre-processing task that involved exporting the glossaries in the PDF and filtering out everything but the relevant tokens.

4.1.2.2 The OOV Annotator Itself

In UIMA, an annotator is any tool that "[does] the real work of extracting structured information from unstructured data". The OOV Annotator would accept raw student writing and capture all out-of-vocabulary or incorrectly spelled words, providing suggestions if applicable.

After some research, I found an open source spellchecker in the DKPro Core set of tools, called Jazzy Spellchecker, upon which I would base my work. Jazzy provided, for example, methods by which one could create a hashmap representation of a word list from a text file. This was crucial for quick dictionary lookup of words in a student response. What I would implement myself would be a nuanced rule-based method for checking misspellings, as well as the suggestions to go along with each type of misspelling. The textbooks glossary contained several hundred proper nouns, I chose to digitally represent and check separately from the other words.

Some of the errors that the implemented Spelling Annotator caught and provided suggestions for:

- Grammar-based capitalization errors (beginnings of sentences, quotes, after colons, etc)
- Capitalization errors for both common and proper nouns
- Misspellings (caught based on the metric of Jazzy's Levenshtein-distance-based cost function; sensitivity of the function would later be calibrated through testing)

Figure 1: The OOV Annotator in the UIMA Annotation Viewer

In the backend of the online workbook, the pipeline would work in the following way: a text of a student's open response would be tokenized and passed through a part of speech tagger, then this token would be passed to the OOV annotator. The lemma of that token would be checked in the dictionary in a hashmap lookup. If the lemma did exist in the dictionary, the raw form of the token would be checked for capitalization errors. If the did not exist in the dictionary, the raw token's string similarity to dictionary terms were checked, to determine if a suggestion could be given. Tokens that were deemed neither in-vocabulary nor incorrectly spelled were then marked as being out of vocabulary. Drawing on my knowledge from a Algorithms and Data Structures course, I aimed to keep the implementation of the tool as computationally efficient as possible.

4.2 LanguageTool: A good idea?

Around the time I was ready to integrate the Spelling Annotator into the main workbook, my colleagues became aware of LanguageTool, a full-service, open-source proofreading tool that

could be applied to a website's text boxes [8]. It was valuable because it offered sophisticated and user-friendly suggestions on all aspects of English style. I was asked to ascertain whether LanguageTool might be a option for the FeedBook Project.

In reading through the documentation, I found a number of potential complications. We were using Java 6 for our Java-to-JavaScript framework Google Web Toolkit (GWT), for example, where LanguageTool required Java 8 or later. Further concerns included the fact that there was no simple way to integrate LanguageTool into a GWT application, that it might clash with OOV Annotator, and that it was disseminated as a public service with accompanying service restrictions.

My colleagues assured me that domain-specific information, such as the out-of-vocabulary checker I developed, and general language well-formedness information, such as provided by LanguageTool, were both equally valuable. With some further research, we successfully incorporated LanguageTool directly into the project's Javascript. Students would from then on have access to a much broader range of suggestions when composing open-style answers in the online workbook.

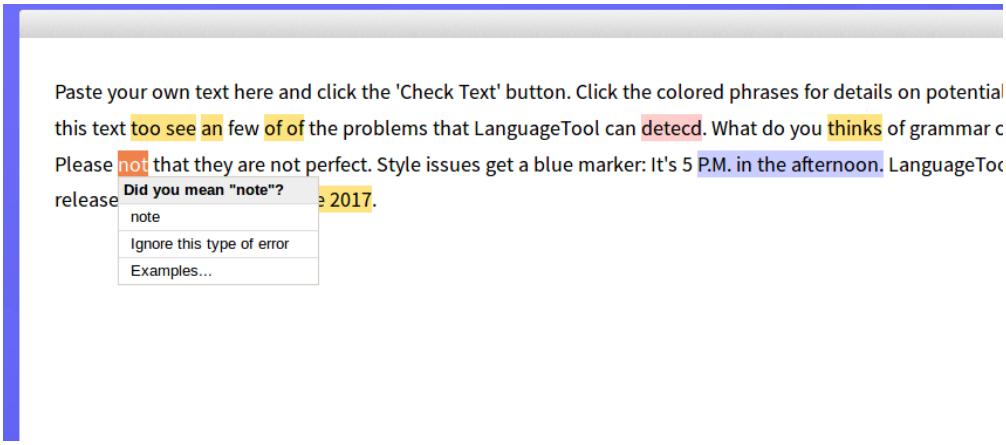

Figure 2: LanguageTool on a text box

4.3 Testing and Debugging

The next step was to integrate the Spelling Annotator into the main project, so that it could be tested on real student responses in the projects database. It was crucial that there were no false positives (i.e. instances in which the annotator incorrectly marked words misspellings or out-of-vocabulary terms). By hand, I had to load and inspect the 63 available open-style responses in the UIMA Annotation Viewer for both false positives, as well as errors that had slipped through the rule-based detection system.

It became clear that the DKPro lemmatizer was not returning the correct lemmas (base forms) of comparative adjectives, which was crucial to check if words were in the dictionary, as dictionary entries are lemmas themselves. For example, "biggest" was lemmatized to "biggest" rather than "big," "worse" to "worse" rather than "bad". Only "big" and "bad" could be found in the dictionary, causing comparative adjective forms to be overlooked. There was also was a similar issue in determining the lemmas of adverbs. Words such as

"quickly" and "happily" were not lemmatizing to "quick" or "happy". I implemented a separate Java class to correct for the pitfalls of the DKPro Lemmatizer, and used it in my OOV Annotator.

In this testing phase, it also came to my attention that many words used in the workbook are not in the textbook's glossary. Unfortunately, the workbook had no glossary of its own, so I compared all OOV terms in student responses to the text in the corresponding exercises. Over 150 new in-vocabulary tokens were found, which I added to the dictionary hashmap.

After these fixes, the OOV Annotator performed in a highly sensitive and highly specific manner; all errors were caught on the set of test documents, and no false positives were given.

4.4 Frontend Integration

The user interface of the online workbook was written using Google Web Toolkit (GWT), a framework which allows Java developers to create and maintain Javascript applications. I had had a small amount of experience with GWT from a previous class. Since it was deemed by my colleagues too complicated from the websites architecture to have suggestion pop-ups hovering over mistakes in text boxes, I added a button that, upon being clicked, would yield a box containing all of a responses OOV terms/misspellings with their respective suggestions. The button and box were necessary to link users with the capabilities of OOV Annotator.

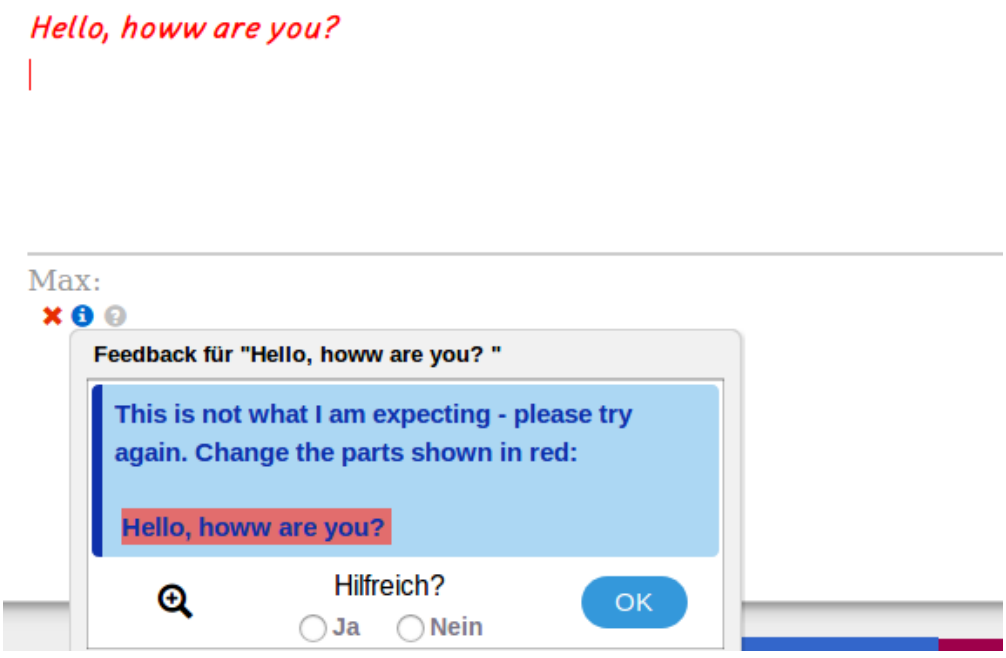

Figure 3: Just after adding the button and box, but before connecting it with the OOV tool

The button would trigger the action of running the answer text through the annotator. The pop-up box, meanwhile, would provide a space where the annotator's output could be displayed. Format multiple suggestions in the same feedback box was one issue that had

to be overcome. To generate the suggestions to be fed into the popup, I learned how to supply server-side processing of text by integrating the OOV Annotator into the projects main pipeline. The annotator was then ready to be used in the online workbook!

5 Discussion

As any developer, I desired more data to generate final performance statistics on my tool. Unfortunately, however, it had been necessary to use the entire small set of available test data to capture new vocabulary terms during the debugging phase. Student data from Feedbooks pilot classrooms was not available to me at the time.

When I gave a final presentation to my colleagues, it was noted that the OOV annotator did not capture two-term dictionary entries such as phrasal verbs such as "put...on" or "turn...off". Given more time, I may have experimented with finite state transducers for this purpose of recognizing such long-distance dependencies.

My internship at the Feedbook Project was a worthwhile experience. I learned a great deal about collaborating on a large codebase with other developers. Personally, I believe there was one aspect of the internship that could have greatly improved my internship. The fact that the project could only be used on a desktop in a separate building meant that I could only describe issues via email, causing long delays for answers about the projects rather complicated architecture. I feel that working in the same space as more experienced colleagues is invaluable for a novice programmer, and I will look for this environment in future positions.

Nevertheless, I am deeply grateful to the team at the Feedbook Project for allowing me to complete my internship with them.

References

[1] Truscott, John. *The case against grammar correction in L2 writing classes*, Language learning 46.2 (1996): 327-369.

[2] Lee, Icy. *Error correction in L2 secondary writing classrooms: The case of Hong Kong.* Journal of Second Language Writing 13.4 (2004): 285-312.

[3] Radecki, Patricia M., and John M. Swales. *ESL student reaction to written comments on their written work.* System 16.3 (1988): 355-365.

[4] *The Feedbook Online Workbook*, available at `http://www.sfs.uni-tuebingen.de/feedbook/`.

[5] *Apache UIMA*, available at `https://uima.apache.org/`.

[6] *uimaFit*, available at `https://uima.apache.org/uimafit.html`.

[7] *DKPro Core*, available at `https://dkpro.github.io/dkpro-core/`.

[8] *Language Tool*, available at `https://languagetool.org/`.